DOCUMENT RESUME

ED 086 223 IR 000 075

AUTHOR Ripota, Peter
TITLE A Concept For a Primary Author's Language (PAL-X)
INSTITUTION Freiburg Univ. (West Germany).
PUB DATE 73
NOTE 20p.; Paper presented at the International Conference on Training for Information Work (Rome, Italy, November 15-19, 1971)

EDRS PRICE MF-$0.65 HC-$3.29
DESCRIPTORS *Computer Assisted Instruction; *Computer Programs; *Documentation; Higher Education; Machine Translation; Program Descriptions; *Programing Languages; *Translation
IDENTIFIERS COURSEWRITER II; LIDIA; PAL X; *Primary Authors Language

ABSTRACT

A Primary Author's Language (PAL-X) has been developed to serve as a documentation language for computer-assisted instructional (CAI) programs. Its development was necessary to permit the dissemination of CAI given the facts that: 1)existing CAI programs were written in over 60 languages; 2)the system for COURSEWRITER II, the most commonly used language, was discontinued by International Business Machines (IBM); and 3)an exchange of programs was found to be nearly impossible. PAL-X was built to contain answer-processing functions, to be independent of hardware and of machine, programing or author languages, to have simple, internationally recognized symbls, and to be capable of partition into subsets. The language was designed to function with non-generative and generative CAI; operations, names and text were separated and the verbosity of programing languages avoided. It was constructed to be used for primary authoring of CAI, to date it has been employed to author CAI material at the University of Freiburg, to translate programs from COURSEWRITER to LIDIA, to introduce students to the principles of CAI, and to document a program in general chemistry. (PB)

FILMED FROM BEST AVAILABLE COPY

ED 086223

A Concept for a Primary Author's Language

(PAL-X)

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

by Peter Ripota

Projekt CUU, Universität Freiburg, W-Germany

Goal

Taking a look into Helen Lekan's "Index to Computer Assisted Instruction" (1) we find about 1300 programs for CAI written in approximately 60 different languages. Making a trend-analysis for the next (fourth) edition, we shall see some 1750 programs plus or minus 200. Both in the second and in the third edition, the most widely used language was COURSEWRITER II, a language running on a system no longer maintained by IBM. The point I'm driving at is the following: How shall this vast amount of instructional material ever be made available to the general public when an exchange of programs (even those written in the same language) is nearly impossible? What is standard procedure in computing science - the publishing of algorithms in a suitable documentation language (ALGOL) (2), or their exchange through a generally available programming language (FORTRAN) - seems to be impossible within the field of CAI. As a result, instructional programs usually are of pure quality due to the lack of competition, criticism and 'survival of the fittest'.

To achieve this there are two ways:

(1) Persuade everybody that APL is the finest language in the world so that everybody writes his program in this language, after which they are freely exchangeable.

IR000075

(2) Develop a documentation language especially for CAI, then start documenting instructional programs and publish these documents.

Since I am not so sure about the feasibility of number one, I pursued number two - trying to create a language suitable for documenting programs in the field of Computer Assisted Instruction. The result is a Primary Author's Language, PAL-X, the 'X' standing for the sub-part of CAI it is intended for. PAL-X is artifical, machine-independent and not intended for immediate computer implementation. It is for the 'man in the field', the teacher who needs some means of expressing his ideas about a certain program in a concise, complete and unambiguous way.

Some of the requirements of such a language are the following:

(1) It is to be developed for purposes of Computer Assisted Instruction (CAI) which means, among other things, that it contain answer-processing functions.

(2) Operations are to be independent of any hardware configuration, machine language, programming language or author language. They are to reflect the logical structure of an instructional program.

(3) Symbols should be simple, self-explanatory and effective.

(4) Symbols for mathematical and logical operations should be as close to international practice as possible.

(5) It should be constructed in such a way that it can be partitioned into independent subsets according to the needs and facilities of existing author languages.

Let me clarify the concept of documentation. People still think that drawing a flow-chart is sufficient to tell programmers and users what the program is about. It is - for the most simple question-answer-type programs so abundant in CAI (and so bad to its reputation). But when you try to document either a program with much text, or with a more complex teaching strategy (e.g. a simulation-game), or with faradvanced algorithms (for analysis of natural language), then a flow-chart is a nice illustration of perhaps some esthetic value. It is by no means sufficient for a programmer trying to understand the program's logic or to implement it on another system. For this one needs a language.

From our few remarks about the goal of the language, its possible applications are easily to be seen:

(1) PAL-X may be used for primary authoring of CAI-programs, making possible a division of labour between author and programmer. Authors thereby are free from burdening their minds with such irrelevant things as loading counters, calling the calculator through some special commands, or other implementation-dependent format-intricacies that have nothing to do with program logic or education. This "source program" in PAL-X can be given to an experienced programmer acting as a human compiler by translating PAL-X-source-code into any convenient programming language - even assembly language to gain effectivity. (This translating process may be done by a mechanical preprocessor; in these manuals, however, we adopt the author's point of view and do not worry about the actual implementation of our ideas.)

(2) PAL-X is, as already said, a means of documenting instructional programs in a complete and unambiguous yet machine independent and easily readable way. It could help tear down the language barrier in CAI and enhance the exchange and evaluation of Computer Assisted Material.

(3) PAL-X is an attempt at a standardisation for CAI and related fields. General books on computing science usually contain examples either in FORTRAN or in ALGOL (or some modified version of these). Nothing comparable is possible in CAI. An author attemping a text-book on CAI would have to write his examples in COURSEWRITER III or APL, since by now these are probably the most widely used author languages.

But COURSEWRITER and APL are very bad documentation languages as everybody knows who tried to read the printout of a CW- or APL-program of moderate complexity. On the other hand, PAL-X is rather short (due to its notational philosophy), covering many fields, operations and applications (due to its independence of actual realisation) and easily expandable to cover aspects of CAI unknown in the past (and present) but of probable great importance for the future, if CAI wants to have any chance of survival (list processing, for example, and formula manipulation).

Content

Adopting the scheme of UTTAL (3) there are three kinds of CAI:

(a) degenerate CAI where the student's reaction is confined to pressing a button (and where you really don't need computer);

(b) selective CAI where all answers have to be prestored by the author;

(c) generative CAI where the possible answers (and in some systems even the questions) are generated by some internal algorithm.

Most existing CAI programs belong to (b) and although they to a great degree are responsible for CAI's bad reputation, they have their justification even in the future. So, PAL-X starts with operations suitable for selective CAI, for which the 'X' is replaced by an 'S':

PAL-S for selective CAI

PAL-S is treated in (4a) and (4b), (4a) offering a gradual introduction to both CAI and programming for people not experienced in any field, whereas (4b) gives a more formal and complete description.

(4c) gives a description of desirable graphic input- and output facilities; hence 'X' now becomes 'G':

PAL-G for graphic input and output

(4d) offers a notation for the documentation of algorithms useful in generative Computer Assisted Instruction; and this subset of the language is called

PAL-A for algorithms.

In this article, we only explain the main features of PAL-S.

Structure

The general principle in constructing PAL was a separation of operations (denoted by special symbols), names (represented by mnemonic constructs in capital letters), and text (i.e. character strings used for output- and input-analysis). The verbosity of common programming languages was avoided in order to get a notation very similar to that of APL in certain ways. Let me give three simple examples.

One of the most important operations in procedure-oriented programming languages is branching. There is a very convenient and natural sign first introduced in APL, the arrow (→). Based on this concept, PAL provides the following operations:

→ label	branch to designated label
→	branch to next frame (where frame will be defined in our final example)
↑	branch back to waiting point after wrong answer (for the concept of 'waiting point' see our third example)
↑↑	branch back from subroutine
↓↓	indicator for reentry point when branching back from subroutine
↓	branch out of program (e.g. forced termination of dialog)

A second example shows how PAL takes into consideration special needs of CAI. Instructional programs make use of counters for counting answers of certain kinds (correct, wrong, unanticipated, within certain limits of tolerance, partially correct, partially wrong, etc.) Assuming the counter-variable x to be incremented by n (n being any positive or negative integer, whereby x is decremented in the latter case), one writes

ad n/c 7	in COURSEWRITER

which is short but not very natural (by which we mean close to mathematical notation), and

```
x:=x+n                    in ALGOL,
```

which is more like mathematics but rather long. PAL introduces two slashes to enclose the variable and the incrementor:

```
/x,n/                     in PAL,
```

meaning the same as the above. If n happens to be 1, then our expression reduces to

```
/x/
```

which is probably the most concise way to express a frequently used operation.

Our third example concerns the concept of the waiting point and the combination of signs reflecting the combination of concepts. The waiting point is that position in the program where the computer is ready to accept input from the terminal. In FORTRAN, this is indicated by the command READ; in APL it's the quad (□), but many CAI-languages provide no special statement which easily leads to confusion. PAL makes use of a small circle (o) hinting at the fact that the program now is open for the student's input:

```
o                         wait (indefinitely) and accept
                          input from terminal
```

Sometimes an author wants to limit input either through space or through time. Let's take the latter case first. Suppose there should be a limit of twenty seconds (as perhaps required in a psychological test); then this is written as

o 20 sec	wait no more then 20 seconds

Suppose the author wants the input to consist of not more than five characters. Then he writes

o 5	accept input with no more than five characters

Finally, in placing a slash over the waiting point, followed by a time appointment, the author indicates that the program waits (for a fixed length of time) but no longer accepts any input:

ø 30 sec	wait thirty seconds but accept no input ("pause")

Our final example will introduce some other important features.

Example

Suppose we have a program about genetics. The student is asked what he should do with a culture of Eschericha coli after certain preparations. The answer should be:

"I put it on a medium without adenine."

or anything semantically equivalent to it.

We want to determine if the student's answer contains the keywords medium, adenine and without (with any of its synonyms). If it does not, we want to check the past performance and take other actions that will be described in the example. A program to achieve our goals would have the following code:

```
1     q16
2      "What will you do with the culture?"
3      A = medium
4      B = no|not|without|free|deficient|minus|-
5      C = ad<enine>
6      o
7      A & B &  C: "Right!", /x/, →
8      A & B & ~C: "Which medium is missing?", /y/, ↑
9      A & ~B & C: "With or without adenine?", /y/, ↑
10     A: "Which medium?", /y/, ↑
11     A: /y/,
12    hint
13     "Choose one of the following media: ...", ↑
14     histidine: "No, that's the wrong medium." /y/, ↑
15     ...
16     ϕ: "I didn't understand you. Try again.", /z/, ↑
17     ϕ: z>5 v & (q13 ... q15)ϕ: →remedy
18       intro:"Think about what you learned in the introduction.", /z/, ↑
19       ~intro & ~A(2): "I'll give you a hint.", /z/, → hint
20       ~: "Still wrong.", /z/, ↑
21     ϕ: "Better read the books first.", /z/, ↓
22    q17
23     "Next question. ..."
24     o
       ...
```

Example (line numbers are for reference only)

Explanations

Line 1 contains the label (q16) that heads the frame. It can be used for branching or as a reference point when checking performance data. It is underlined to distinguish it from other operations.

Line 2 contains the text that is to be printed (indicated via the quotation marks " and ").

In line 3 we start to define our keywords. In this way we can always write "A" when we talk about "medium", which makes the combination of keywords through logical connectors much easier.

B is defined as all the synonyms for "without" meaningful in this context. The synonyms are separated by a bar (|) (this is called alternation) and the letter B (called identifier or name) stands for any one of the words in line 4.

C represents the word "adenine" but the program only checks for the leading characters "ad", the letters"denine" being unnecessary ('no-care characters'). They are included only for the information of the programmer or for whoever reads the program.

The symbol in line 6 is the waiting- or reentry point. The program stops execution and waits for a student input.

Line 7 through 21 show answer processing. Let's start with line 7.

If the student's answer contains all three keywords (i.e. "medium", a synonym from B and "ad") the student gets a confirmation, 1 is added to the variable x that counts the correct answers (/ / is the counting operator) and the program branches to the next frame - that is, to the label immediately preceding the next waiting point. Since the next waiting point is in line 24, the next frame starts at line 22 with the label q17.

If C is missing (line 8), then the student is given a hint, the variable y (or the counter y) that counts the wrong answers is incremented by 1, and the program branches back to the reentry point in line 6. A similar process takes place in line 9 where B is missing in the student's answer. Line 12 contains a new label which we'll ignore for the moment (it is ignored by the program when it jumps from one line to the next). Line 10 and 11, both starting with A:, show the automatic sequence control of PAL-S: The first time the student answers "A" (i.e., "medium" without "B" and "C") he hits line 10, is given a hint and branched back to the reentry point to give a new answer. The second time he says "medium" (without "B" and "C") he gets the hint at line 13 where a list of media is printed for him. One of this media is "histidine", and if he gives it as an answer he is transferred to line 14 where the usual actions take place.

(Line 15 signifies other prestored answers (media) we are not interested in.)

The overstruck o ("ϕ") in line 16,17, and 21 denotes an unanticipated answer, that is any character string (or none at all). Student is transferred to line 16 when all matches with prestored answers or their combinations fail the first time; he is transferred to line 17 the second time his answer is unanticipated, and to line 21 the third time this unfortunate event takes place. z is the counter that counts the unrecognized (unanticipated) answers.

Line 17 through 19 show how performance data can be used.

The first question the program asks itself is: Was the number of unanticipated answers (z) greater than 5 or was there an unanticipated answer (ϕ) at every label from q13 till q15 inclusive? ("every" is indicated by the unary use of the logical and (&), meaning all in this case, and q13 ... q15 means: look for the condition at the indicated range of labels. Underlining of labels is unnecessary because of the range indicator ...).

If the condition in line 17 proves to be true, the program branches to the label remedy; if the condition proves to be false, everything at the right

of the colon (:) is ignored and execution continues at the next line. (Note that right of the branching operator (→) labels need not be underlined since branching always takes place to a label.)

Assuming that our condition in line 17 proved 'false', the program looks at the next condition in line 18 and checks, if during program execution label intro was encountered at least once. If that is the case, a hint is printed referring to that frame, z is incremented by one and the student is branched back to the reentry point at line 6.

If the condition "intro" proved 'false' the program would continue at the next line.

In line 19, the program checks if frame intro was not encountered and answer "A" was not given a second time. If this is true, the hint of line 13 was not yet given to the student and a branch to the label hint (to present this hint) is meaningful.

If noneof the above conditions proved true (indicated by a unary "~" in line 20, corresponding to else in other programming languages) a not very stimulating text is printed and the student is branched back to the reentry point.

Lines 18 to 20 showed a procedure called "nesting" in computing jargon. These lines are only executed if the first condition in line 17 ("ϕ") proved true. Otherwise they are ignored and the program skips directly to line 21, where, if a third unanticipated answer was given, a throw-out occurs.

At line 22 a new frame starts.

Application

PAL-X has been successfully applied to primary authoring of instructional material at Projekt CUU (Computer-unterstützter Unterricht = Computer-Assisted Instruction) at the University of Freiburg;

to translating one CAI-program (PFLABE), developed in Freiburg, from COURSEWRITER via PAL-S into LIDIA for implementation at the Zentralstelle für Programmierten Unterricht in Augsburg, whereby the original COURSEWRITER-program remained unknown to the translater in Augsburg;

to introducing students into the principles of Computer-Assisted Instruction at the Computing Centre of the University of Köln (two two-hour courses for two semesters);

and to documenting a program on general chemistry (CUS-Chemie), which to the author's knowledge is the first attempt at a complete and machine-independent documentation of a CAI-program (see ref. (5)).

Appendix

The appendix lists the important commands and procedures in PAL-S

Commands for input, output and answer analysis

" "	quotation marks to indicate text to be printed or displayed.
o	waiting- or reentry point. Encounter of this symbol causes the program to stop and wait for an input.
& or ∅ (a blank)	denotes catenation ("and", "both", "all of them")
v or |	denotes alternation ("or", "and/or", "at least one of them")
~ or ¬	denotes negation ("not")

: or =>	denotes implication ("if - then", "do what is to the right if the condition on the left is true")
ϕ	denotes an unanticipated answer or any character string (which may be the 'empty word', too) (any-string sign or phi)
@	denotes any one character (any character sign)
A, B, C,...	names for string variables. They always start with a capital letter, and they may contain up to eight letters or letters plus digits.
< >	enclose characters that may be missing (are not checked for) in an answer, but are useful information for whoever reads the program.

Special variables, functions and procedures in PAL-S

LEN(a ... b)	number of characters in the answer between a and b inclusive; either a or b or both may be missing
INPUT	a buffer containing the student's latest input; may be referenced by position, e.g. INPUT(7); INPUT(2 ... 5) etc.
POS(a,n)	position of n-th appearance of a in INPUT. Default n: n = 1
EQU1	upper case letters are converted to lower case letters
EQU2	lower case letters are converted to upper case letters
EDIT(a)	eliminates a in INPUT wherever it appears
EDIT(a=b)	replaces a by b in INPUT wherever a appears
EDF	same as EDIT, but with first appearance only
STRICT	keyword-matching is replaced by character-by-character matching
SEQ	sequence of keywords in INPUT has to correspond to sequence of keywords in prestored logical expression
SYN	matches numbers and formulas semantically
KEY(n)	input has to contain at least n of the specified keywords to achieve a match
PHON	phonetic encoding of answer
CALC	to allow or forbid the student's use of the computer as a desc-calculator

Branching in PAL-S

→label	branch to specified label
→	branch to next frame
↑	branch to reentry point
↑↑	branch back from subroutine
↓↓	reentry indicator for branch back from subroutine
↓	branch out; termination sign

Checking performance history in PAL-S

&(range of labels): ...	all-quantifier. Check to see if every label of the indicated range was encountered during program execution
v(range of labels): ...	existence-quantifier. Check to see if at least one of the indicated labels was encountered during program execution
&(lab1 ... lab10) A: ...	see if answer A was given at every label from lab1 till lab10
v(lab1 ... lab10) A: ...	same as above, but any label
lab: ...	check to see if label lab was encountered during program execution

Literature

(1) LEKAN, Helen A.: Index to Computer-Assisted Instruction.
Third Edition, Harcourt Brace Jovanovitch 1971

(2) Communications of the Association for Computing Machinery (CACM), Algorithms Section

(3) UTTAL, W.R., et al.: Generative Computer Assisted Instruction in Analytic Geometry.
ENTELEK Inc., Newburyport 1970

(4) RIPOTA, P.: A Concept for a Primary Author's Language (PAL).
Projekt CUU, Universität Freiburg, 1971 - 1973

a) PAL-S - User's Guide
b) PAL-S - Reference Manual
c) PAL-G - Graphic Input- and Output Facilities
d) PAL-A - Generative Computer Assisted Instruction

(5) GEIST, W.; RIPOTA, P.: CUS-Chemie. Eine maschinenunabhängige Dokumentation. Projekt CUU, Universität Freiburg 1973